FINAL TECHNICAL REPORT

Contract Information

Principal Investigator:	Professor Aloysius K. Mok
PI Institution:	University of Texas at Austin
PI Phone:	(512)471-9542
PI E-mail:	mok@cs.utexas.edu

Grant Title:	Formal Design Methodology for Hard-Real-Time Systems
Grant Number:	N00014-89-J-1472
Grant Period:	October 1, 1988 to May 31, 1994
ONR Scientific Officer:	Dr. Gary Koob

1. Numerical Productivity Measures

- Refereed papers submitted but not yet published: 2
- Refereed papers published: 35
- Unrefereed reports and articles: 13
- Books or parts thereof submitted but not yet published: 0
- Books parts thereof published: 2
- Patents filed but not yet granted: 0
- Patents granted: 0
- Invited presentations: 16
- Contributed presentations: 0
- Honors received: 12
- Prizes or awards received: 2
- Promotions obtained: 1
- Graduate students supported: 8
- Post-docs supported: 1
- Minorities supported: 1

COLLEGE OF NATURAL SCIENCES

THE UNIVERSITY OF TEXAS AT AUSTIN

Department of Computer Sciences • Taylor Hall 2.124 • Austin, Texas 78712-1188 • (512) 471-7316

May 9, 1995

Defense Technical Information Center
Building 5, Cameron Station
Alexandria, Virginia 22314

Dear Sir,

Enclosed please find a copy of my final technical report for Office of Naval Research grant no. N00014-89-J-1472. I am submitting this report in compliance with the requirements for closing out this grant.

Sincerely,

Aloysius K. Mok

Associate Professor

2. Technical Results

2.1 Highlight of results

The goal of this project is to investigate the fundamental scientific issues that will provide a foundation for a formal design methodology for hard-real-time systems. We think that it is fair to say that the progress made by this group parallels and indeed defines much of the direction and advance in real-time software research in the last 10 years.

To bring a scientific discipline to real-time system design, we must develop formal techniques

- to specify the real-time behavior of systems (our contributions: RTL, RTCTL, and APTL)
- to query and validate the desired behavior of a design (our contributions: RTL deductive system, RTCTL model checker, APTL tableau verifier)
- to ensure that a design can indeed be implemented by proper resource allocation algorithms (our contributions: solutions to adaptive real-time task scheduling, a fast admissibility test for sporadic tasks, complexity results and algorithms for the pinwheel scheduling problem)
- in addition, we need to demonstrate that our theory can be translated into design tools that form an integral part of a design methodology (our contributions: the Modechart suite of tools)

We have obtained a number of basic results in all these areas and also expanded the horizon of real-time computing in the direction of

- rule-based systems (our contributions: the EQL, MRL real-time rule-based programming environment)
- semantics and concurrency control of real-time data (our contributions: the similarity relation and its use in real-time scheduling)

We discuss these results below.

2.2 Technical Results

2.2.1 Fundamental results in RTCTL and RTL

In [4], we invented a system of temporal logic RTCTL (Real Time Computation Tree Logic) which extends CTL (Computation Tree Logic), a logic that has been widely applied to reasoning about program correctness. RTCTL melds qualitative temporal assertions together with real-time constraints to permit specification and reasoning at both qualitative and quantitative levels of abstraction. The technical device we used to model time is a simple time metric on a computation path, specifically the number of state transitions. Syntactically, this is accomplished by augmenting the usual CTL modal operators with an integer parameter k such that assertions can now be made to express bounded-time eventuality, e.g., assertions of the form: some event must happen within k

steps on all possible computation paths leading from the current state. We developed a model-checking algorithm for RTCTL which, like CTL, has complexity linear in the size of the RTCTL specification formula and in the size of the global state-space graph. The key observation that makes this possible is that the model-checking algorithm actually recovers not only whether an eventuality is fulfilled, but also when, based on calculating its rank in the Tarski-Knaster sequence of approximations. This is joint work with E. A. Emerson and his group.

In contrast to temporal logic, RTL is a first-order-logic and uses no modal operators. As one may expect, the expressiveness of first-order-logic-based specification methods often leads to the undecidability of the underlying logic. An interesting problem is therefore to investigate the decidability of subsets of RTL which are nonetheless sufficiently powerful to express practical real-time system requirements. Four basic results were obtained. The first result concerns a subset of RTL which does not have OR operators for which we showed the existence of a high-order polynomial decision algorithm. This algorithm is related to the decision problem of the emptiness of context-free languages.

The second result is a negative result. We showed that the unrestricted use of occurrence indices, even when the occurrence arguments do not contain arithmetic operators, is sufficient to lead to undecidability. The proof is via reduction from 2-counter machine to RTL formulus.

The third result stemed from an attempt to remedy the second (negative) result. We came up with a reasonable restriction on the usage of occurrence indices and showed the existence of a doubly exponential decision algorithm for a subset of RTL under that restriction. This restriction fits in properly with the semantics of mode transitions in our Modechart language for specifying the control structure of real-time system. In the course of this investigation, we also developed an extension of RTL for modeling asynchronous real-time systems which allows explicit use of "local" clocks. This is documented in [25].

The fourth result is a by-product of the third result. We used the same technique for deriving the third result to develop a temporal (propositional) logic for asynchronous real-time system which has a much simpler decision procedure than previous approaches. This algorithm has the same asymptotic time complexity as the decision procedure in the third result. This is documented in [22].

2.2.2 Logic for asynchronous distributed real-time systems in APTL

The goal for this work is to answer the following question. In a distributed system with multiple local clocks, how do we specify timing assertions with respect to the local clocks in a unified way and still retain elementary decidability? For motivation, consider the following scenario.

A pizza parlor has the following policy. "Any pizza order through the phone that is not delivered within 30 minutes will be served free." Technically, there is a serious ambiguity with the statement of this policy. Specifically, on whose clock is the "30 minutes" measured ? Is the pizza parlor allowed to stop its clock every time before a deadline expires, or is the customer allowed to use her/his own watch that ticks twice as fast as the pizza parlor's clock? The usual "implicit" assumption is that the customer's and the

server's clocks are precisely synchronized. But what if this assumption does not hold (as is in the case of geographically distributed systems such as space-borne defense systems)?

The synchronized clock assumption has been adopted implicitly in previous research by Alur and Henzinger who used traditional (synchronous) inequality operators ($\leq$, $=$) to compare time values as read on different local clocks. They treated local clock readings as numbers. The truth value of an inequality comparing separate local clock values is determined by the usual axioms of integer arithmetic. Suppose we are in a pizza parlor and two clock readings were observed:

PizzaOrder = 400 according to the customer's local clock, and
PizzaDelivered = 430 according to the pizza parlor's local clock.

Then the assertion "PizzaOrder+30 = PizzaDelivered" is true because 400+30 = 430. Alur and Henzinger actually showed that using the synchronous (arithmetic) inequality for comparing readings of different clocks renders the extension of their logic TPTL undecidable. Thus, it seems to be hopeless to retain elementary decidability in the case of reasoning about distributed real-time systems, at least in the context of propositional temporal logic.

In our work, we proposed a different interpretation of inequalities involving separate local clocks. In some respects, the following two formulas are not equivalent.

(1) PizzaOrder+30 = PizzaDelivered
(2) 400+30 = 430

First, formula (1) provides more information than formula (2) because we know that PizzaOrder=400 is a reading of the customer's clock and PizzaDelivered=430 is a reading of the pizza parlor's clock. This type of information is completely lost in formula (2). Second, formula (1) does not necessarily make sense. Since the two clocks may tick at completely independent rates, why should we compare the numerical values of their readings ? Furthermore, in formula (1), it is unclear on which clock the duration "30" is measured. In other words, formula (1) really compares "apples" (ticks of the customer's clock) with "oranges" (ticks of the pizza parlor's clock).

Our solution is based on the following principles. (1) Local clock readings are treated as special temporal markers instead of numerical values. (2) Readings of different clocks are distinct from each other. (3) Inequalities comparing different local clock readings are interpreted by means of "asynchronous inequality operators". The definition of asynchronous inequality operators is designed to preclude comparison of local clock readings on different sites against an absolute (global) time scale. This allows us to skirt the undecidability result of Alur and Henzinger. We have applied these ideas to generalize TPTL from the single clock model to the multiple clock model. The result is a language, APTL (Asynchronous PTL) for the specification and verification of hard real-time asynchronous systems ([22]).

APTL is based on an asynchronous system model which permits definition of

inequalities asserting the temporal precedence of local clock readings on different sites. This logic can be used to specify and reason about such important properties as bounded drift rates of local clocks. We gave a tableau-based decision procedure for determining APTL satisfiability. This procedure has double exponential time complexity in the size (number of bits) of the input formula. we implemented an APTL verifier by incorporating a number of optimizing heuristics into the tableau-based procedure. This verifier passed the benchmark programs: asynchronous railroad crossing example (20 sec.) and simplified HARM missile example (40 sec.).

To further gain performance, we also worked out a symbolic model checking algorithm for distributed real-time systems [26]. Unlike other approaches, we treat timing predicates and atomic propositions uniformly by translating everything into the mu-calculus. This allows us to use BDD (Binary Decision Diagram) directly for model-checking a real-time logic such as TCTL without resorting to extraneous techniques such as term-rewriting. we completed an implementation of this BDD-based technique for the synchronous version of APTL which we call Synchronous Real-time Event Logic (SREL). This implementation can be used to verify a modechart specification against a timing property written in SREL. Our algorithm first constructs a symbolic trace graph to capture the semantics of a given modechart and then performs symbolic model checking on the trace graph. As a part of the algorithm, we proposed an efficient BBD encoding scheme for solving the time representation problem. This is an important contribution since the utility of BDDs for timing verification has been in doubt because of the potential for exponential blow-up in the BDD representation of timing properties. Our encoding scheme produces BDDs whose size are provably O(n) for timing inequalities involving n-bit time values. We are able to do this by observing that the arithmetic inequalities of interest in real-time systems have at most two variables (comparing two event occurrences) and involve only "counting" operations.

We incorporated our BDD encoding scheme for time representation into the SMV system (version 2.3) from Carnegie-Mellon University. For the benchmark programs that we have run our augmented version of SMV on, we have been able to achieve one to two orders of magnitude in speedup and space saving when compared to the direct implementation of timing and event counting functions by integer operations provided by the unaugmented SMV system. The details of these results are described in the paper "Symbolic Model Checking for Event-Driven Real-Time Systems" which will appear in RTSS '93.

2.2.3 Load adjustment in adaptive real-time systems

Most extant work in real-time resource allocation assumes that a set of real-time processes have fixed timing parameters, e.g., periods, deadlines, minimum separation. The resource scheduling problem is to determine whether a scheduler exists (the feasibility problem) or whether a given scheduling policy can meet the timing requirements of the set of processes (the schedulability problem). Although much progress has been made in regard to both the feasibility and schedulability problems, there are relatively few results in the case where the processes may change their timing parameters in reaction to the environment. Such cases may occur for a number of reasons: (1) A process may be used to monitor a physical object which demands different amounts of attention

depending on the state of the external world. (2) A process may be created or deleted in reaction to the arrival or departure of a physical object and there is no complete a priori knowledge of the objects. (3) A process may be reallocated (appears as a new process) to run on a different processor because another processor has broken down. In our work, we showed that the real-time resource allocation problem can be reformulated to model the adaptive nature of the processes so as to handle dynamic objects (case 1), overload recovery (case 2) and fault tolerance (case 3). In particular, we introduced the "configuration selection problem" and gave a PTIME solution to a variant of the problem ([17]).

To formulate the scheduling problem where timing parameters may vary with time, we introduce the notion of an "adjustable periodic process" which is a set of pairs: {(c_1, p_1), ... (c_k, p_k)}. Each pair in this set is called an "adjustment candidate" of the adjustable periodic process. Each adjustment candidate of a adjustable periodic process P is a candidate for load adjustment, i.e., we can select any one of the pairs in P to be used as input timing parameters to the run-time scheduler. A configuration C of a set M of n dynamic periodic processes is a set of n pairs {(c_1, p_1), ... (c_n, p_n)} such that each pair is an adjustment candidate of exactly one of the dynamic periodic processes in M. Each configuration represents a set of periodic processes with fixed parameters. Not all configurations are schedulable. The configuration selection problem is to select among a list of configurations one which is is schedulable.

To tackle the configuration selection problem, we have sharpened a result due to Liu and Layland concerning a schedulability test of periodic processes by a fixed priority scheduler. Specifically, two process transformations are introduced for the purpose of deriving a bound for the achievable utilization factor of processes whose periods are related by harmonics. This result is then generalized so that the bound is applicable to any process set, and an efficient algorithm to calculate the bound is provided. Based on this result, we presented a solution to the configuration selection problem under the restriction that a fixed priority scheduler is used at run time and task utilization must not exceed the achievable utilization factor. We further showed that a generalization of the configuration selection problem (the period assignment problem) is NP-hard and also gave efficient approximation algorithms for solving this problem by exploiting known algorithms for solving the set covering problem.

2.2.4 Scheduling of Sporadic Task Systems

Sporadic tasks were introduced by Mok in his Ph.D. dissertation to model external interrupts. Clearly one cannot guarantee scheduling for interrupts that can occur arbitrarily frequently. The idea of assuming a minimum delay between invocations was introduced so that scheduling guarantees might be made while retaining a useful model of something in the real world. In cases where this assumption is not realistic, the run-time scheduler must be able to deduce when a sporadic task request cannot be met. This is trivially accomplished by monitoring when a deadline is actually missed, but by then it may be too late to perform certain overload recovery procedures. So the question is whether we can detect an overload condition way before it happens. Ideally, we would like to detect whether the addition of a sporadic task request can lead to overload at the time the request arises. Our collaboration with L. Rosier's group resulted in necessary

and sufficient conditions for a sporadic task system to be feasible. These conditions cannot, in general, be tested efficiently (unless P = NP). They do, however, lead to a feasibility test that runs in efficient pseudo-polynomial time for a very large percentage of sporadic task systems. This opens up the possibility of a practical algorithm for early detection of processor overload and for dynamic load adjustment. This work is reported in [11].

Our investigation of the periodic maintenance scheduling problem was originally motivated by our study of sporadic tasks scheduling. A sporadic task is characterized by an ordered triple $T=(c,d,p)$ where c is the required computation time and d is the deadline measured from request time, and p is the minimum separation. If t and t' are two invocation times, then $|t-t'| \geq p$. For example, if $T=(1,10,10)$ then T may be invoked at times 8 and 18, but not at times 8 and 17. In hard real-time systems, we must deal with worst-case combinations of sporadic task invocations. This is generally done by transforming them into periodic tasks such that a schedule for the transformed system is also a schedule for the original system. Mok gave one such transformation in his Ph.D. dissertation. We presented a paper [5] which answered a number of the decision problems related to the simplest case of the sporadic task problem: the pinwheel scheduling problem. It was pointed out to us by Professor C. L. Liu that in the case of dense task sets, our problem is similar to that of periodic maintenance. In earlier work, Liu gave a sufficient but not necessary condition for an instance of the periodic maintenance problem to be schedulable. We extended Liu's work on periodic maintenance in several major ways.

The k-server, $k \geq 1$, periodic maintenance problem consists of n "machines" each of which must receive maintenance at fixed intervals from one of k servers each of which can service only one machine at a time. A ''slot'' is defined as the time needed for one server to service one machine. A k-server schedule S for an instance $B = \{b_1, b_2, \ldots, b_n\}$ is an infinite sequence of k-multisets over $\{1,2,\ldots,n,blank\}$ where *blank* can be repeated within a multiset but $1,\ldots,n$ cannot, such that successive occurrences of i are exactly b_i slots apart. In [5], we provide a comprehensive complexity analysis of the decision problems concerning periodic maintenance. We show that the k-server periodic maintenance scheduling problem is NP-complete in the strong sense; that the periodic maintenance schedule verification problem CoNP-complete in the strong sense. We also obtained positive results in the following categories. When each maintenance interval is a multiple of all shorter intervals, the decision problem is in PTIME. When an instance consists of only two distinct numbers, the decision problem is in PTIME. For both categories, we showed that the single-server scheduling problem is in S-P-C, i.e., for each instance of these problems, there exists a PTIME program generator whose output is a single-server scheduler that operates in constant time per scheduling decision made. This result can be generalized to the multiple server case for the first category. This is joint work with L. Rosier and his group.

2.2.5 EQL and Response Time Analysis:

An EQL program has a set of rules for updating variables which denote the state of the physical system under control. The firing of a rule computes a new value for one or more state variables to reflect changes in the external environment as detected by sensors.

Sensor readings are sampled periodically or generated by interrupts. Every time sensor readings are taken, the state variables are recomputed iteratively by a number of rule firings until no further change in the variables can result from the firing of a rule. The equational rule-based program is then said to have reached a fixed point. Intuitively, rules in an EQL program are used to express the constraints on a system and also the goals of the controller. If a fixed point is reached, then the state variables have settled down to a set of values that are consistent with the constraints and goals as expressed by the rules. EQL differs from the popular expert system languages such as OPS5 in some important ways. Whereas the interpretation of a language like OPS5 is defined by the recognize-act cycle, the basic interpretation cycle of EQL is defined by fixed point convergence. We believe that the time it takes to converge to a fixed point is a more pertinent measure of the response time of a rule-based program than the length of the recognize-act cycle.

Even though the problem of deciding whether a EQL program has bounded response time is decidable whenever there is a finite state-space representation for the program, we can show that the decision problem is PSPACE-hard [7]. To combat the combinatorial state space explosion, we have developed a general iterative strategy to determine if a EQL has bounded response time. This strategy works by identifying independent subsets of rules in the EQL program, matching these subsets with special forms and if successful, rewriting the program to make use of the fact that some variables have been established to be stable. The power of this general strategy obviously depends on the generality of the special forms employed. We have developed three more special forms. Special form B has been programmed into our analysis tool, and will be described here to illustrate this work. First, we need a few definitions.

L_a = { v | v is a variable that appears in the LHS of rule a }
R_a = { v | v is a variable that appears in the RHS of a rule a }
T_a = { v | v is a variable that appears in the enabling condition of rule a }

Given two rules a and b, let x_i, i=1,...,n be variables in the intersection of the sets L_a and T_b. Rule a is said to **potentially enable** rule b if the assignment statement in rule a may assign values the variables in $\{x_i\}$ such that there is at least a combination of values for the variables in T_b but not in L_a which enables rule b.

The enable-rule (ER) graph of a set of rules is a labelled directed graph G=(V,E) where V is a set of vertices each of which is uniquely labelled by a rule number i, and E is a set of edges such that there is an edge from vertex a to vertex b if rule a potentially enables b.

A set of rules are said to be in special form B if all of the following conditions hold: (1) For every rule, only constant terms are assigned to all the variables in L. (2) All the rules are compatible pairwise (i.e., either their enabling conditions are mutually exclusive or they must not assign different values to the same variable). (3) For each cycle in the ER graph of the rules, no two rules that appear in the same cycle may assign different values to the same variable, even if their enabling conditions are mutually exclusive.

We have proved that a EQL program whose rules are in special form B will always reach a fixed point in a bounded number of steps. We have also developed an efficient

algorithm for checking whether a set of rules are in special form B which runs in $O(MAX(n^2, m \times MAX(n, e)))$ where n and e are respectively the number of vertices and edges in the ER graph. Special form B was needed to analyze the NASA Space Station modules that MITRE provided to us. This is documented in [8].

To enhance the applicability of our analysis tool, we completed the design of Estella, a facility for EQL programmers to customize the analyzer by specifying application-specific knowledge. Estella was used to verify the boundedness of the response time of some NASA applications (see [13]).

Significant advance was also obtained in work on the more advanced language MRL which uses EQL as its "core language". We have improved the fixed point detection and timing analysis algorithms MRL so that it can be used to handle those MRL programs with non-constant assignments. This is achieved by extending the AM (access-modify) graph representation to include semantic information of the rules. The extension preserves all the properties, including the the "transfer principle" which is vital to our analysis algorithms. The speed of the fixed point detection algorithms was improved dramatically by designing a much more efficient compatibility detection algorithm which has been a bottleneck in the performance factor ([20]).

In applying MRL tools to an application from NASA Johnson Space Center, we discovered an inevitable limitation of the fixed-point detection algorithm. The limitation is due to the lack of knowledge of users' assumption on input data. Therefore, we undertook to improve the fixed-point detection algorithm by allowing users to specify application-specific information about the program which would otherwise be unobtainable. We have included two types of semantic information: functional dependency and data pattern specification. This work is reported in [19].

2.2.6 Semantics and concurrency control of real-time databases

While past work in database performance stress mostly on throughput, there is increasing interest in the performance of transaction systems that have significant response time requirements. These requirements are usually posed as hard or soft deadlines on individual transactions, so that a concurrency control algorithm must attempt to meet deadlines as well as preserve database consistency. There have been a number of analytic and simulation studies on the performance of scheduling algorithms to meet deadlines. In these studies, database consistency is preserved by enforcing serializability. However, our interaction with Navy personnel has convinced us that serializability is often too strict a correctness criterion for real-time applications, where the precision of an answer to a query may still be acceptable even if serializability is not strictly observed in transaction scheduling. Obviously, violation of serializability must be justified in the context of the semantics of the application domain. We explored a weaker correctness criterion for concurrency control in real-time transactions, namely, the notion of "similarity" [23].

Similarity is a binary relation on the domain of a data object. Every similarity relation is reflexive and symmetric, but not necessarily transitive. Different transactions can have different similarity relations on the same data object domain. Two views of a transaction are similar iff every read event in both views uses similar values with respect to

the transaction. We say that two values of a data object are similar if all transactions which may read them consider them as similar. In a schedule, we say that two event instances are similar if they are of the same type and access similar values of the same data object. We say that two database states are similar if the corresponding values of every data object in the two states are similar. A minimal restriction on the similarity relation that makes it interesting for concurrency control is the requirement that it is preserved by every transaction, i.e., if a transaction maps database state s to state t and state s' to t', then t and t' are similar if s and s' are similar. The concept of similarity is used to extend the usual correctness criteria for transaction scheduling: final-state, view, conflict serializability to their counter-parts of final-state Delta-serializability, view Delta-serializability, and conflict Delta-serializability. The main difficulty in these extensions is that similarity is in general not a transitive relation. This poses a limit on how events can be swapped in a schedule and complicates the usual notion of equivalence among schedules. Several technical results are needed to justify a notion of equivalence that permits a restricted form of event swapping.

It has been recognized by many researchers that the notion of serializability is too strict a correctness criterion for concurrency control in accesssing real-time data. In avionics software, for example, the precision of an answer to a query involving sensor data is often acceptable as long as the data is sufficiently timely, even though updates are sometimes performed in violation of the usual serializability criterion. Instead of read/write locks, a "cyclic executive" is routinely used in these applications to enforce a set of timing constraints on data access which in turn guarantees data consistency, the usual database locking protocols being too inefficient for the purpose. Obviously, violation of serializability must be justified in the context of the semantics of the application domain. In our RTSS92 paper, we introduced the concept of similarity which is a binary relation on the domain of a data object. We reported how the similarity concept can be used to justify the legality of the non-serializable access of sensor data common in cyclic executives.

Having thus provided a semantic foundation for accessing real-time data, an obvious question is how the notion of similarity relates to real-time transaction scheduling. Intuitively, we expect significant gains in scheduling flexibility since more write events can be deferred and some read events can be scheduled AHEAD of the latest write events. We started investigating quantitatively how the additional flexibility in read/write event scheduling made available by similarity considerations may relax the real-time scheduling problem. In a paper which appeared in RTSS '93, we examine a class of scheduling policies which we call Similarity Stack Potocols (SSP) whose correctness is justified by similarity. The SSP protocols can significantly improve the schedulability of real-time transaction workloads on multiprocessor systems. We have obtained both analytical utilization bounds for the SSP protocols as well as demonstrated their real-time performance advantages in simulation experiments. We use as our workload the real-time benchmark suite, "realistic" numbers from the NRL software model for the HARM missile as well as statistically generated task sets. Our performance studies are fundamentally different from past reports on analytic and simulation results for meeting transaction deadlines in that unlike previous studies, we do not assume serializability as the correctness criterion for database consistency.

We have also completed the implementation of a preliminary version of a Real-Time Object Management Interface package which utilizes some of these ideas on the Intel iRMK real-time kernel.

2.2.7 The Modechart Design Language and Verifier

The methodology for verifying properties of systems specified in Modechart was first presented in the paper "A Method for Verifying Properties of Modechart Specifications", published in Proceedings 9th RTSS, Huntsville, AL, 1988. That paper introduceds a method for verifying certain types of RTL formulas with respect to modechart specifications, based on the notion of a computation graph. A computation graph can be viewed as the modechart analog of a reachability or invariance graph. The technique described in the paper uses two steps. First the computation graph for the given modechart is generated, and then the decision procedure appropriate to the type of formula being verified is applied to the graph to determine if the formula is valid with respect to the graph (and so for the specification).

There were three major areas that were not addressed by the 1988 paper. The first of these is a problem known as "splitting". The problem arises when trying to determine which events are associated with a point in the computation graph. The solution (which gives the problem its name) is to split the children of a point based on whether or not they inherit the events of their parent. The second problem is known as the disabling problem. This problem deals with how to enforce the timing constraints imposed by transitions that have been disabled before their deadlines. The solution is to impose the deadlines of disabled transitions on the transitions that disable them. The two problems are related because splitting also imposes additional timing constraints on points in the computation graph. The disabling problem also provides the basis of the technique for dealing with modecharts with sophisticated hierarchical structure. Two other loose ends leftover from the 1988 paper are practical implementation issues for formula verification, and formal proofs for some of the main results, in light of the new solutions to the splitting, disabling, and hierarchical problems above. All of these are addressed in a paper that appeared in 1990 Real-Time Systems Symposium [10].

The first version of the first generation verifier was implemented in June of 1989. Since that time the initial program has been considerably enhanced and updated to include the splitting and disabling solutions mentioned in the preceding paragraph. Since then, numerous improvements have been made. Some of these improvements and the algorithms used in the verifier, are the subject of the paper "Implementing a Verifier For Real-Time Systems" which appeared in RTSS90.

Some general semantic issues that concern Modechart are discussed in the paper "Clairvoyance, Capricious Timing Faults, Causality, and Real-Time Specifications" [18]. The first deals with the semantics of timing constraints, and the second with the semantics of event constraints. The usual method of enforcing timing constraints in the semantics of a specification is by simply declaring that all timing constraints are satisfied. This approach breaks down in the presence of timing constraints that can not be met, in particular, it can result in the need for clairvoyance on the part of an implementation to avoid such constraints in order to accurately reflect the formal semantics. The solution to the

problem is for the semantics to allow computations that encounter unsatisfiable timing constraints, but which halt on doing so. This not only eliminates the need for clairvoyance, but also produces more realistic behavior and allows for reasoning about the behavior of the specified systems in the period leading up to the failure.

The second area of concern is an apparent loss of causality due to event constraints in the absence of interleaving. If the semantics of the specification do not use interleaving to model concurrency, so that all events which occur at the same time are treated as simultaneous, and if an event can be caused by an event simultaneous with it, it may be possible for an event to cause itself. As an example, consider a modechart transition with its own transition event as its triggering condition. At any time the transition occurs it is enabled, and at any time it does not occur it is not enabled. This is a degenerate example in that the circularity is immediate and unavoidable, but this need not be the case. Also, even the degenerate example possesses computations that obey the intuitive notion of causality. Those computations of such a system where there is a loss of causality are termed non-linearizable, because the events in the computation can not be totally ordered in a way such that each event is caused by events strictly before it. The paper provides methods both for generating non-linearizable computations and for modifying the syntax and semantics of a specification language to avoid them.

Both of these issues were first encountered in investigating properties of the Modechart specification language. In particular the atomicity of actions leads to the necessity for clairvoyance if unsatisfiable timing constraints are to be avoided. Likewise, as previously indicated, certain transition conditions can give rise to non-linearizable computations.

The other Modechart related work has been to clarify, correct, and justify several aspects of the semantics and syntax that were unclear or incorrect, largely in response to queries from the A-7 group at NRL, in particular Bruce Labaw and Connie Heitmeyer. This has been reflected in part in Paul Clement's new Modechart Quick Reference Guide [47]. Substantial implementation effort was undertaken in response to NRL requests and falls into three broad categories. The first category is to improve the performance and user interface of the current program. Accordingly, the current version of the verifier, "verifyw" now accepts the new standard ascii modechart format, and has been integrated into Paul Clements "modechart" program, a combined user interface that integrates all of the current SARTOR/Modechart software. Similarly, the current version of the program has been restructured to simplify modifiability, and to enable all of the routines to use a more efficient clustering algorithm for calculating distances. The second category is to increase the range of Modechart specifications which the verifier can accept as input. The current version of the program has been extended so that almost all legal transition conditions have been implemented, as have actions and state predicates. In the process of doing so, three technical problems associated with actions and certain types of transition conditions have been addressed, and are discussed in the memos "Adding Actions to Computation Graphs" and "Angle Brackets and the Tile Problem". The last category is actual functionality increases. Verifyw now can check for persistent points, busy-wait modes, and implements three special cases of two interval formulas for mode inclusion and exclusion which can be checked efficiently. Preliminary work in attempting to generate constant values (bounds) in timing constraints and formulas has also been done,

some of which has been implemented, as requested by NRL personnel.

3. Research Transitions and DoD Interactions

3.1 Naval Research Laboratory

We have consistently collaborated with the Naval Research Laboratory in the evaluation of Modechart and Modechart-related tools (especially the verifier), and in the implementation of the advanced Modechart prototype. Numerous trips were made by Paul Clements to the Naval Research Laboratory where researchers there are exploring the use of Modechart in a Navy embedded hard-real-time engineering environment. NRL personnel have been briefed on the re-implementation effort, and are continuing joint investigations. Their contributions have included, to date, software to implement other abstract data types useful in Modechart processing (sets, lists, modes), and more importantly, the sponsoring of an exhaustive survey of object-oriented data base management systems that would be appropriate to host a re-tooled Modechart environment.

NRL has also investigated methodological issues associated with using Modechart in a real development environment. Their preliminary work, which is being explored in conjunction with UT personnel, addresses how a "canonical" Modechart for a hard-real-time embedded system such as an avionics program should be structured for maximum flexibility. In cooperation with NRL, we have drafted a specification of a Navy embedded real-time software system using Modechart. The system is the HARM Low-Cost Seeker (HLCS) [29], a version of the Navy's HARM anti-radar missile with upgraded avionics, developed at China Lake, California. HLCS is an example of a state-of-the-art, parallel-processing, hard-real-time, embedded, *actual* Navy system, and the modeling effort is designed to test Modechart's usefulness on a non-contrived application. Modechart's scaleability, its ability to model systems with a priority-based pre-emptive scheduler, the ability of its supporting analysis tools (verifier, simulator) to deal with problem spaces of nontrivial size, and methodological issues involving management of complexity are all being evaluated.

The Modechart simulator has also been used in the HLCS specification effort. This marks the first time that the simulator has been used in a "production" capacity, and reviews of its performance are very positive. The Modechart simulator now has action semantics added to its repertoire. Work on the new scenario language, which may be considered a new front end specification language for Modechart, continues apace. A fully automatic scenario-to-Modechart translator has been implemented, and is being tested and checked out on three completely real-world examples (the A-7 aircraft pilot data panel protocols, the engine start procedures for the P-3 Orion aircraft, and the signal processing software for HLCS, described above).

In 1979 the Naval Research Laboratory published a requirements specification for the A-7E aircraft. This specification relied on finite-state machine (represented as tables) to define the outputs, as a function of system state, required by the avionics software. The method was found to have clearly defined completeness and consistency rules, and in many ways has made the task of specifying real-time software much easier. The requirements specification was one of the most successful (and widely transferred) products of NRL's Software Cost Reduction project, and recent work at UT has identified an algorithm for translating NRL-style tabular specifications into modecharts.

The NRL style is better suited for displaying certain types of finite-state- based requirements; the equivalent modechart representation is much less concise. Therefore, by adding the option of specifying modecharts in the NRL form, users are provided with an input paradigm better suited and more concise for certain types of requirements. However, because we can translate automatically into Modechart and therefore into RTL, none of the semantic properties of Modechart is lost. We have therefore added the semantic rigor of Modechart to the expressiveness of the NRL format, and enriched the Modechart environment.

3.2 Naval Weapons Center, China Lake

P. Clements visited the Naval Weapons Center (NWC) at China Lake, California, in April and July of 1991. While there, he consulted with personnel to learn about requirements and requirements specification problems for hard-real-time embedded systems that are currently fielded by the Navy. He spoke with Bob Westbrook (Avionics), John Seybold (HARM Low-Cost Seeker), Ken Trieu (F/A-18), Bob Page (Missile Software), and others. As a result of his visit, an experiment was initiated at UT to apply Modechart-related tools, especially the design of a "scenario specification" language to the engine-start procedure of the Navy's P-3 Orion aircraft.

A. Mok and P. Clements visited the Naval Weapons Center (NWC) at China Lake, California, in April of 1992. While there, Mok gave 4 hours of lecture to NWC personnel concerning recent advance in our research. He consulted with Navy personnel to discuss real-time system design issues. Clements discussed his work on the HARM with NWC personnel.

3.3 Industrial collaboration

In technology transfer to industry, we have received support and collaborated with personnel of Texas Instruments Corporation. R. H. Wang visited TI and ported some of our timing analysis tools for rule-based programs to the Central Research Laboratory of TI at Dallas. Wang was able to apply the tools to analyze the Threat Assessor program, a threat assessment expert system for the D/NAPS system presented to him on the spot. The Threat Assessor program uses fuzzy logic to compute a lethality value of a threat to an aircraft. The boundedness of the response time of this program was verified.

Two rule-based modules of the Space Station Expert System: (1) the Integrated Status Assessment Expert System (ISA), and (2) the Fuel Cell Monitoring Expert System (FCE) were obtained from MITRE and translated into EQL. A report of our analysis has been written and we have been invited by MITRE to give a presentation of our results.

As a move towards experimentation of full-scale real-time expert systems, we collaborated with Browne et al [35] in the implementation of the MERCURY expert system language. MERCURY is modelled after the experimental language MRL which we have developed in collaboration with Professor D. Miranker's group at the University of Texas Advanced Research Laboratory (ARL). A prototype of MERCURY was completed and it is being used by software engineers at ARL to develop distributed tactical simulation systems for the U.S. Army.

We also initiated a collaboration effort with General Electric Corporation. We designed a simplified diagnostic model [45] for the local reactivity controller of the Advanced Liquid Metal Reactor plant (a key commercial development of the GE Advanced Reactor Programs) through discussing with GE engineers. This model was programmed in MRL and analyzed by our timing analysis tools. GE engineers were sufficiently impressed that arrangement is being made for us to apply our tools to a significant subsystem of their project. Unfortunately, the GE effort was discontinued when they failed to obtain funding from DOE to continue the work.

4. Software and Hardware Prototypes

Modechart:

Our Modechart verifier has been integrated with the toolset "MT", carried out in collaboration with NRL. The synergistic capability of MT is described in the paper "MT: A Toolset for Specifying and Analyzing Real-Time Systems" which will appeared in RTSS '93. MT is available from NRL.

Timetool/TAL:

Timetool/TAL is an interactive general code analyzer for estimating the minimum and the maximum execution time of the input source code. Previously, we have only a Sun window user interface for Timetool. Under this grant, we built an X window user interface for Timetool. This implementation will contribute greatly to the portability of this tool. We are still receiving requests for the timing analysis tool set: timetool/tal which we developed in 1989.

EQL response time analyzer:

A timing analyzer for EQL has gone through several generations of improvement and will be released to the general public through internet access.

5. Awards and Honors

- A. Mok is the recipient of a $100,000 Faculty Fellowship at the University of Texas at Austin for the years 1989-1994.

- A. Mok served on the program committees of 12th Real-Time Systems Symposium, San Antonio, December 1991, 9th IEEE Real-Time Operating Systems and Software Workshop, Pittsburg, May 1992, Second International Workshop on Responsive Computer Systems, Tokyo, October 1992, 3rd Workshop on Responsive Computer Systems, New Hampshire, October 1993, 13th Real-Time Systems Symposium, Phoenix, December 1992, 1992 International Computer Symposium, Tai-Chung, Taiwan, December 1992.

- A.K. Mok was elected Vice Chairman of the IEEE Technical Committee on Real-Time Systems, 1993-1994.

- A. Cheng obtained his doctorate in computer science in August, 1990.

- S. Sutanthavibul obtained his doctorate in computer science in May, 1991.

- C. K. Wang obtained his doctorate in computer science in May, 1992.

- Farn Wang obtained his doctorate in computer science in May, 1993.

- Sanjoy Baruah obtained his doctorate in computer science in August 1993.

- Paul Clements obtained his doctorate in computer science in May 1994.

6. Publications, Presentation and Reports

1. "A Method for Verifying Properties of Modechart Specifications", F. Jahanian and D.A. Stuart, Proceedings of the 9th Real-Time Systems Symposium, Huntsville, December 1988.
2. "The Pinwheel: a Real-time Scheduling Problem", R. Holte, A. Mok, L. Rosier, I. Tulchinsky and D. Varvel, Proceedings of the 22nd Hawaii International Conference on System Science, January 1989.
3. "Evaluating Tight Execution Time Bounds of Programs by Annotations", A. Mok, P. Amerasinghe, M. Chen and K. Tantisirivat, Proceedings of the Sixth Workshop on Real-Time Operating Systems and Software, Pittsburgh, May 1989.
4. "Quantitative Temporal Reasoning", E.A. Emerson, A. Mok, J. Srinivasan and A.P. Sistla, Proceedings of Workshop on Automatic Verification Methods for Finite State Systems, Grenoble, France, June 1989
5. "Algorithms and Complexity of the Periodic Maintenance Problem", A. Mok, L. Rosier, I. Tulchinsky and D. Varvel, Proceedings of EUROMICRO Symposium 89, Cologne, September 1989.
6. "Multiprocessor On-line Scheduling of Hard-Real-Time Tasks", M. Dertouzos and A. Mok, IEEE Transactions on Software Engineering, vol. 15, no. 12, December 1989.
7. "Formal Analysis of Real-Time Equational Rule-Based Systems", A. Mok, Proceedings of the 10th Real-Time Systems Symposium, Santa Monica, December 1989.
8. "Fast Textual Analysis of Real-Time Rule-Based Systems to Verify Their Fixed Point Convergence," A. Cheng and C. K. Wang, Proceedings of 5th Annual Conference on Computer Assurance COMPASS-90, June 1990.
9. "Bounded-Time Fault-Tolerant Rule-Based Systems", J.C. Browne, E.A. Emerson, M. Gouda, D. Miranker, A. Mok, L. Rosier, Telematics and Informatics, vol. 7, no. 3/4 pp. 441-454, 1991
10. "Implementing a Verifier for Real-Time Systems", D.A. Stuart, Proceedings of the 11th Real-Time Systems Symposium, Orlando, December 1990
11. "Preemptively Scheduling Hard-Real-Time Tasks on One Processor", S. Baruah, A. Mok and L. Rosier, Proceedings of the 11th Real-Time Systems Symposium, Orlando, December 1990.
12. "MRL: A Real-Time Rule-Based Production System", C. K. Wang, A. Mok and A. Cheng, Proceedings of the 11th Real-Time Systems Symposium, Orlando, December 1990.
13. "Estella: A Facility for Specifying Behavorial Constraint Assertions in Real-Time Rule-Based Systems", A. Cheng, J.C. Browne, A. Mok and R.H. Wang, Proceedings of 6th COMPASS Conference, Gaithersburg, Maryland, June 1991.
14. "Semantics of Real-Time Database Systems", A. Mok and T.W. Kuo, Proceedings of 1991 ONR/INRIA Joint Workshop on Responsive System Design.
15. "Engineering CASE Tools to Support Formal Methods for Real-Time Software Development", C. Heitmeyer, B. Labaw, P. Clements and A. Mok, Proceedings, Fifth International Workshop on Computer-Aided Software Engineering, Montreal, Canada, July 6-10, 1992.
16. "Formal Specification of Asynchronous, Distributed Real-Time Systems in APTL",

F. Wang, A. Mok and E.A. Emerson, Proccedings, 14th International Conference on Software Engineering, Melbourne, Australia, May 11-15, 1992.

17. "Load Adjustment in Adaptive Real-Time Systems", T.W. Kuo and A. Mok, Proceedings of Real-Time Systems Symposium, San Antonio, December 1991.
18. "Clairvoyance, Capricious Timing Faults, Causality, and Real-Time Specifications," Douglas A. Stuart and Paul C. Clements, Proceedings of Real-Time Systems Symposium, San Antonio, December 1991.
19. "Automated Analysis of Bounded Response Time for Two NASA Expert Systems", C.K. Wang, D.C. Tsou, R.H. Wang, J.C. Browne and A. Mok, Proceedings of IEEE SIGSOFT'91 Conference, New Orleans, December, 1991.
20. "Timing Analysis of MRL: a Real-Time Rule-Based System", C.K. Wang and A.K. Mok, Real-Time Systems Journal, vol. 5, no. 1, March 1993, pp. 89-128.
21. "Quantitative Temporal Reasoning", E.A. Emerson, A.K. Mok, A.P. Sistla and J. Srinivasan, Real-Time Systems Journal, vol. 4, no. 4, December 1992, pp. 331-352.
22. "Distributed Real-Time System Specification and Verification in APTL", F. Wang, A.K. Mok and E.A. Emerson, ACM Transactions on Software Engineering and Methodology, October 1993, vol. 2, no. 4, pp. 346-378.
23. "Application Semantics and Concurrency Control of Real-Time Data-Intensive Applications", T.W. Kuo and A.K. Mok, Proceedings of Real-Time Systems Symposium, Phoenix, Arizona, December, 1992.
24. "Deriving Response-Time Bounds for Equational Rule-Based Programs", R.H. Wang and A.K. Mok, Proceedings of 1992 International Computer Symposium, Taichung, Taiwan, December, 1992.
25. "Asynchronous Real-time Event Logic", F. Wang and A.K. Mok, Proceedings of 1992 International Computer Symposium, Taichung, Taiwan, December, 1992.
26. "Symbolic Model Checking for Distributed Real-Time Systems", F. Wang, A.K. Mok and E.A. Emerson, Proceedings of 1993 Formal Method Europe Conference, Denmark, April, 1993.
27. "Applying Formal Methods to an Embedded Real-Time Avionics System", P. Clements, C. Heitmeyer and A.K. Mok, Proceedings of the 1993 IEEE Workshop on Real-Time Applications, Manhattan, New York, May 1993.
28. "A Verifier for Distributed Real-Time Systems with Bounded Integer Variables", F. Wang and A.K. Mok, Proceedings of the 1993 IEEE COMPASS Conference, Maryland, June, 1993.
29. "MT: A Toolset for Specifying and Analyzing Real-Time Systems", P. Clements, C. Heitmeyer, B. Labaw and A.T. Rose, Proceedings of the 14th IEEE Real-Time Systems Symposium, Raleigh-Durham, North Carolina, December 1993.
30. "SSP: a Semantics-Based Protocol for Real-Time Data Access", Tei-Wei Kuo and Aloysius K. Mok, Proceedings of the 14th IEEE Real-Time Systems Symposium Raleigh-Durham, North Carolina, December 1993.
31. "Symbolic Model Checking for Event-Driven Real-Time Systems", J. Wang, A.K. Mok and F. Wang, Proceedings of the 14th IEEE Real-Time Systems Symposium, Raleigh-Durham, North Carolina, December 1-3, 1993.
32. "Correct and Robust Decision Systems for High Complexity Critical Control Systems", J.C. Browne, E.A. Emerson, M. Gouda, D. Miranker, A.K. Mok, S. Chodrow, R.H. Wang, D. Tsou and L. Obermeyer, Proceedings of Third International

Workshop on Responsive Computer Systems, Lincoln, New Hampshire, September 29-October 1, 1993.
33. "Using Data Similarity to Achieve Synchronization for Free", T.W. Kuo and A.K. Mok, Proceedings of 11th IEEE Workshop on Real-Time Operating Systems and Software, May 1994.
34. "RTL and Refutation by Positive Cycles", F. Wang and A.K. Mok, Proceedings of the Formal Methods Europe'94 Conference, Barcelona, Spain, October, 1994; LNCS 873, Springer-Verlag.
35. "A New Approach to Modularity in Rule-Based Programming", J.C. Browne, E.A. Emerson, M.G. Gouda, D. Miranker, R. Bayardo Jr., S. Chodrow, D. Gadbois, F. Haddix, T.W. Hetherington, A.K. Mok, L. Obermeyer, D.C. Tsou, C.K. Wang and R.H. Wang, Proceedings of International Conference on Tools in AI, New Orleans, 1994.
36. "Deriving Response-Time Bounds for EQL Programs with Rule Priorities", R.H. Wang and A.K. Mok, 15th Real-Time Systems Symposium, December 7-9, 1994, Puerto Rico.
37. "Safety Analysis of Timing Properties in Real-Time Systems", F. Jahanian and A.K. Mok, in Readings in Ultra-Dependable Distributed Systems, edited by N. Suri, C.J. Walter and M.M. Hugue, IEEE Computer Society Press, 1994.
38. "Formal Specification of Real-Time Systems", F. Jahanian, A. Mok and D. Stuart, Manuscript, Real-Time Systems Group, Department of Computer Science, University of Texas at Austin, 1989
39. "Internal Structure of MRL Execution System", C. K. Wang, Manuscript, Real-Time Systems Group, Department of Computer Science, University of Texas at Austin, October, 1991
40. "A Quick-Reference Guide to the Modechart Model and Semantics", P. Clements, Manuscript, Real-Time Systems Group, Department of Computer Science, University of Texas at Austin, 1991
41. "A Customizable Fixed-Point Convergence Analyzer for Real-Time Rule-Based Systems", Rwo-Hsi Wang, Master's Thesis, Dept. of Computer Sciences, Univ. of Texas at Austin, December 1991.
42. "Reducing The Size of eqc Output", Rwo-Hsi Wang, Technical Report of Real-Time Systems Group, Univ. of Texas at Austin, August 1992.
43. "Translating the Collision Avoidance System Program into MRL", D.C. Tsou, Technical Report of Real-Time Systems Group, Univ. of Texas at Austin, December 1991
44. "Developing Diagnostic Model for the Local Reactivity Controller of the Advanced Liquid Metal Reactor", Technical Report of Real-Time Systems Group, Univ. of Texas at Austin, July 1992.
45. "Modeling the HARM Low-Cost Seeker System with Modechart," P.C. Clements, B. Labaw, A. Bull, Technical Report of Real-Time Systems Group, Univ. of Texas at Austin, in progress. 46. "Verifyw Quick Reference Guide", D. Stuart, Technical Report of Real-Time Systems Group, Univ. of Texas at Austin, 1992.
47. "CG Module Commentary", D. Stuart, Technical Report of Real-Time Systems Group, Univ. of Texas at Austin.
48. "Abstract Interface Specifications for the Computation Graph Data Structures Module (CG)", P. Clements, Technical Report of Real-Time Systems Group, Univ. of Texas

at Austin, 1992.

49. "An Incremental Arc-Consistency Match Algorithm for MRL", Duu-Chung Tsou, Real-Time Systems Group Technical Report, Dept. of Computer Sciences, Univ. of Texas at Austin, August 1993.
50. "Real-Time Object Management Interface on iRMK", T.W. Kuo, A.K. Mok and D. Chan, Real-Time Systems Group Technical Report in progress, Dept. of Computer Sciences, Univ. of Texas at Austin, August 1993.

Invited Lectures

1. "Timing Analysis of Hard-Real-Time Software"
   Intel Corporation, Oregon, June 1, 1989.
2. "Advance in Real Time Logic and Scheduling"
   Newcastle University/ICL International Seminar on Real-Time Systems, September 5-7, 1989.
3. "Real-Time Scheduling Theory",
   Ada Run-Time Environment Working Group (ARTEWG) meeting, Boston, April 1990.
4. "Specification, Analysis and Synthesis of Hard-Real-Time Systems",
   Brazilian National Computer Science Conference, July, 1990.
5. "Static vs. Dynamic Mechanisms in Critical System Applications",
   Panel presentation in Second International Conference on Dependable Computing for Critical Systems, Tucson, Arizona, February 1991.
6. "Bounded-Time Rule-Based Systems",
   Workshop presentation, 7th IEEE CAIA Conference, Miami Beach, Miami, February 1991.
7. "Coping with Implementation Dependencies in Real-Time System Verification",
   REX Workshop, Department of Mathematics and Computing Science, Technische Universiteit Eindhoven, June, 1991.
8. "Analysis and Synthesis of Real-Time Control Structures in RTL",
   Carnegie Mellon University, Systems Group Seminar, April, 1991.
9. "The MB Paradigm for Fault-Tolerant Real-time Systems Design",
   School and Symposium, Formal Techniques in Real-Time and Fault-Tolerant Systems, 6-10, January 1992.
10. "Coping with Implementation Dependencies in Real-Time Systems",
    Department of Computing Science, Oxford University, January 1992.
11. "Review of Real-time Systems Design Research",
    Naval Weapons Center, China Lake, California, April 15-16, 1992.
12. "Real-Time Systems Specification and Verification",
    Chung Shan Institute of Science and Technology, Taiwan, ROC, June, 1992.
13. "Load Adjustment for Dynamic Real-Time Processes",
    University of Hong Kong, June, 1992.
14. "Conceptual Foundations of Real-Time System Design",
    NATO Advanced Study Institute on Real-Time Computing, Sint Maarten, October, 1992.
15. "What Really Is Rapid Prototyping For Real-Time Systems?",
    AT&T Bell Laboratories, Naperville, Illinois, June 1993.
16. "Resource Bounded Systems",
    The Joint Directors of Laboratories Basic Research and Computer Science Panels, Marcy, New York, June 1993.